Amendment to the Claims:

This listing of claims replaces all prior versions, and listings, of claims in the application:

1. (Currently amended) A method comprising:

receiving N/M machine instructions directing a processor to search an array of N data elements, where N and M are integers greater than one; and

executing a first ~~each~~ machine instruction by:

~~retrieving M data elements in a single fetch cycle;~~

concurrently comparing ~~the~~ M data elements retrieved when executing a previous machine instruction to M corresponding current extreme values; ~~and~~

updating a set of M references based on said comparing; and

retrieving another M elements in a single fetch cycle to be compared when executing a subsequent machine instruction.

2. (Currently amended) The method of claim 1, wherein said retrieving ~~the~~ another M data elements comprises retrieving the another M data elements as a single data quantity containing the another M data elements.

3. (Previously presented) The method of claim 2, wherein the set of M references comprise pointer registers to store addresses of extreme data quantities in the array of N data elements.

4. (Canceled)

5. (Previously presented) The method of claim 1, wherein M = 2 and N is greater than two.

6. (Currently amended) The method of claim 1, wherein executing ~~each~~ the first machine instruction further includes:

storing the current M extreme values in M accumulators; and

copying the M data elements to the accumulators based on said comparing.

7. (Currently amended) The method of claim 5, wherein said concurrently comparing the M data elements comprises processing a first data element with a first execution unit of a pipelined processor and processing a second data element with a second execution unit of the pipelined processor.

8. (Previously presented) The method of claim 5, wherein concurrently comparing the M data elements comprises

concurrently processing a first data element and a second data element within a single execution unit of a pipelined processor.

9. (Currently amended) The method of claim 1, wherein said concurrently comparing the M data elements to M corresponding current extreme values includes determining whether each of the data elements is less than the corresponding current extreme value.

10. (Currently amended) The method of claim 1, wherein said concurrently comparing the M data elements to M corresponding current extreme values includes determining whether each of the data elements is greater than the corresponding current extreme value.

11. (Currently amended) A method for searching an array of N data elements for an extreme value, the method comprising:

issuing N/M machine instructions to a processor, wherein the processor is adapted to process M data elements in parallel;

executing each machine instruction by:

~~retrieving M data elements in a single fetch cycle;~~

concurrently comparing ~~the~~ M data elements to corresponding M current extreme values, ~~and~~

retrieving another M elements in a single fetch cycle to be compared when executing a subsequent machine instruction;

updating accumulators and pointers associated with the M current extreme values based on said comparing; and

analyzing results of the machine instructions to identify at least a value and a position of at least one ~~the~~ extreme value in the array.

12. (Currently amended) The method of claim 11, further comprising:

setting up registers for said accumulators and pointers.

13. (Currently amended) A method comprising:

retrieving a pair of data elements from an array of elements in a single fetch operation, wherein the pair of data elements includes an even data element and an odd data element;

substantially comparing the even element of the pair with an even extreme value;

if the even element of the pair exceeds the even extreme value, storing the even element of the pair as the even extreme value and storing a parameter indicative of a location of the even element of the pair;

concurrent with said comparing the even element of the pair with the even extreme value, comparing the odd element of the pair with an odd extreme value;

if the odd element of the pair exceeds the odd extreme value, storing the odd element of the pair as the odd extreme value; and

substantially fetching and comparing remaining pairs of data elements of the array until all of the data elements of the array have been processed.

14. (Previously presented) The method of claim 13, further comprises setting the even extreme value as a function of the even element of the element pair and setting the odd extreme value as a function of the odd element of the element pair.

15. (Previously presented) The method of claim 13, further comprises maintaining a first accumulator to store a minimum value for the even elements and a second accumulator to store a minimum value for the odd elements.

16. (Currently amended) The method of claim 13, ~~further including~~ wherein storing the parameter indicative of a location of the even element of the pair comprises maintaining a first pointer register to store an address for the extreme value of

the even data elements, and further comprising maintaining a second pointer register to store an address for the extreme value of the odd data elements.

17. (Previously presented) The method of claim 16, further including adjusting at least one of the pointer registers after processing all of the pairs of data elements to account for a number of stages in a pipeline.

18. (Previously presented) The method of claim 13, wherein the method is invoked by issuing N/M machine instructions to a programmable processor, wherein N equals a number of elements in the array, and M equals a number of data elements that the processor can concurrently compare.

19. (Previously presented) An apparatus comprising:

a execution pipeline adapted to process M data elements in parallel; and

a control unit adapted to direct the execution pipeline to search an array of N data elements for an extreme value in response to N/M machine instructions, the execution pipeline being configured to:

retrieve M data elements from the array of N data elements in a single fetch cycle;

concurrently compare the retrieved M data elements to corresponding M current extreme values, and

update accumulators and pointers associated with the M current extreme values based on said comparing.

20. (Previously presented) The apparatus of claim 19, wherein in response to the machine instructions, the control unit directs the pipeline to set up registers for accumulators and pointers.

21. (Previously presented) The apparatus of claim 19, wherein the pipeline includes M registers adapted to store accumulators and pointers associated with the extreme values.

22. (Original) The apparatus of claim 21, wherein the registers are pointer registers.

23. (Original) The apparatus of claim 21, wherein the registers are general-purpose data registers.

24. (Previously presented) The apparatus of claim 19, wherein the pipeline includes M accumulators to store M current extreme values.

25. (Previously presented) The apparatus of claim 19, wherein the pipeline includes M general-purpose registers to store M current extreme values.

26. (Previously presented) An article comprising a medium having computer-executable instructions stored thereon for compiling a software program, wherein the computer-executable instructions are adapted to generate N/M machine instructions to search an array of N data elements to find an extreme value, each machine instruction causing a programmable processor to:

retrieve M data elements from an array of N elements in a single fetch operation;

concurrently compare the retrieved M data elements to M a corresponding current extreme values; and

update accumulators and pointers associated with the M current extreme values based on said comparing.

27. (Canceled)

28. (Original) The article of claim 26, wherein each machine instruction causes the processor to concurrently process a first data element and a second data element within a single execution unit of a pipelined processor.

29. (Previously presented) A system comprising:

a memory device; and

a processor coupled to the memory device, wherein the processor includes a pipeline configured to process M data elements in parallel and a control unit configured to direct the pipeline to search an array of N data elements for an extreme value in response to N/M machine instructions, wherein in response to each of the machine instructions, the pipeline being configured to:

retrieve M data elements from the array of N data elements in a single fetch cycle;

concurrently compare the retrieved M data elements to corresponding M current extreme values, and

update accumulators and pointers associated with the M current extreme values based on said comparing.

30. (Canceled)

31. (Previously presented) The system of claim 29, wherein the pipeline includes M registers configured to store the accumulators and pointers.

32. (Original) The system of claim 31, wherein the registers are pointer registers.

33. (Original) The system of claim 31, wherein the registers are general-purpose data registers.

34. (Original) The system of claim 29, wherein the memory device comprises static random access memory.

35. (Original) The system of claim 29, wherein the memory device comprises FLASH memory.

36. (New) The method of claim 11, wherein the at least a value and a position of at least one extreme value in the array comprises a value and a position of a first occurrence of a minimum value in the array.

37. (New) The method of claim 11, wherein the at least a value and a position of at least one extreme value in the array comprises a value and a position of a last occurrence of a minimum value in the array.

38. (New) The method of claim 11, wherein the at least a value and a position of at least one extreme value in the array comprises a value and a position of a last occurrence of a maximum value in the array.

39. (New) The method of claim 11, wherein the at least a value and a position of at least one extreme value in the array comprises a value and a position of a first occurrence of a maximum value in the array.